

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number  
**WO 01/088696 A3**

(51) International Patent Classification<sup>7</sup>: **G06F 9/50**

(21) International Application Number: PCT/GB01/02170

(22) International Filing Date: 18 May 2001 (18.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
0011974.3 19 May 2000 (19.05.2000) GB

(71) Applicant and

(72) Inventor: SMITH, Neale, Bremner [GB/GB]; 40 Royal Oak, Alnwick, Northumberland NE66 2DA (GB).

(74) Agent: KENNEDYS PATENT AGENCY LIMITED  
Floor 5, Queens House, 29 St. Vincent Place, Glasgow G1 2DT (GB).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,

CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

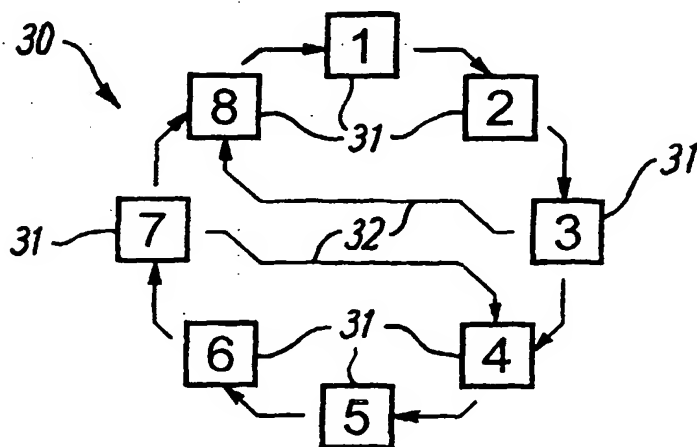
(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published  
with international search report.

(86) Date of publication of the international search report:  
12 September 2002

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: PROCESSOR WITH LOAD BALANCING



(57) Abstract: The present invention relates to a system and method of distributing workload among processors (11) in a multi-processor system (10), with workload being transferred through a plurality of transfers between processor pairs (12), such that the plurality of pairs together define a closed loop. The present invention enables a processor to automatically balance its workload with other similar processors connected to it, with minimal interprocessor connection.

WO 01/088696 A3

# INTERNATIONAL SEARCH REPORT

International Application No

PCT/GB 01/02170

A. CLASSIFICATION OF SUBJECT MATTER  
IPC 7 G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

INSPEC, EPO-Internal

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GEHRKE J E ET AL: "Rapid convergence of a local load balancing algorithm for asynchronous rings" DISTRIBUTED ALGORITHMS. 11TH INTERNATIONAL WORKSHOP, WDAG '97. PROCEEDINGS, vol. 220, no. 1, 6 June 1999 (1999-06-06), pages 1-18, XP002200403 Theoretical Computer Science, Elsevier, Netherlands ISSN: 0304-3975	1-8
Y	page 1, line 1 -page 3, line 32	9-11
Y	US 5 031 089 A (LIU HOWARD T ET AL) 9 July 1991 (1991-07-09) abstract; figure 4	9-11

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

### \* Special categories of cited documents:

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the international filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \*Z\* document member of the same patent family

Date of the actual completion of the international search

28 May 2002

Date of mailing of the international search report

13/06/2002

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Michel, T

## INTERNATIONAL SEARCH REPORT

International Application No

PCT/GB 01/02170

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>CORTES A ET AL: "On the performance of nearest-neighbors load balancing algorithms in parallel systems" PARALLEL AND DISTRIBUTED PROCESSING, 1999. PDP '99. PROCEEDINGS OF THE SEVENTH EUROMICRO WORKSHOP ON FUNCHAL, PORTUGAL 3-5 FEB. 1999, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 3 February 1999 (1999-02-03), pages 170-177, XP010321821 ISBN: 0-7695-0059-5 page 172, right-hand column, line 1 -page 173, left-hand column, line 21</p>	1-11
A	<p>ZAMBONELLI F: "Exploiting biased load information in direct-neighbour load balancing policies" PARALLEL COMPUTING, ELSEVIER PUBLISHERS, AMSTERDAM, NL, vol. 25, no. 6, June 1999 (1999-06), pages 745-766, XP004176773 ISSN: 0167-8191 page 749, paragraph 3.1 -page 751</p>	1,7
A	<p>EP 0 756 233 A (TAO GROUP LTD) 29 January 1997 (1997-01-29) page 3, line 51 -page 4, line 38</p>	1,7
A	<p>NIKHIL R S ET AL: "T: A MULTITHREADED MASSIVELY PARALLEL ARCHITECTURE" COMPUTER ARCHITECTURE NEWS, ASSOCIATION FOR COMPUTING MACHINERY, NEW YORK, US, vol. 20, no. 2, 1 May 1992 (1992-05-01), pages 156-167, XP000277763 ISSN: 0163-5964 page 156, right-hand column, line 29 - line 35</p>	5

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/GB 01/02170

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 5031089	A	09-07-1991	NONE	
EP 0756233	A	29-01-1997	GB 2272085 A	04-05-1994
			EP 0756232 A1	29-01-1997
			EP 0756233 A1	29-01-1997
			AU 679686 B2	10-07-1997
			AU 4508093 A	24-05-1994
			AU 676815 B2	20-03-1997
			AU 6203696 A	10-10-1996
			AU 676816 B2	20-03-1997
			AU 6203796 A	10-10-1996
			CA 2146672 A1	11-05-1994
			DE 69309704 D1	15-05-1997
			DE 69309704 T2	30-10-1997
			DE 69322887 D1	11-02-1999
			DE 69322887 T2	27-05-1999
			DE 69327739 D1	02-03-2000
			DE 69327739 T2	28-09-2000
			EP 0667011 A1	16-08-1995
			WO 9410628 A1	11-05-1994
			GB 2286269 A ,B	09-08-1995
			GB 2293674 A ,B	03-04-1996
			GB 2293675 A ,B	03-04-1996
			HK 1005474 A1	08-01-1999
			HK 1005475 A1	08-01-1999
			HK 1005476 A1	08-01-1999
			JP 8502612 T	19-03-1996
			SG 34219 A1	06-12-1996
			SG 52857 A1	28-09-1998
			SG 42445 A1	15-08-1997
			US 5930511 A	27-07-1999

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number  
WO 01/88696 A2

- (51) International Patent Classification<sup>7</sup>: G06F 9/00 (81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (21) International Application Number: PCT/GB01/02170
- (22) International Filing Date: 18 May 2001 (18.05.2001)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:  
0011974.3 19 May 2000 (19.05.2000) GB
- (84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).
- (71) Applicant and  
(72) Inventor: SMITH, Neale, Bremner [GB/GB]; 40 Royal Oak, Alnwick, Northumberland NE66 2DA (GB).
- Published:  
— without international search report and to be republished upon receipt of that report
- (74) Agent: KENNEDYS PATENT AGENCY LIMITED;  
Floor 5, Queens House, 29 St. Vincent Place, Glasgow G1 2DT (GB).  
For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: PROCESSOR WITH LOAD BALANCING

(57) Abstract: The present invention relates to a system and method of distributing workload among processors (11) in a multi-processor system (10), with workload being transferred through a plurality of transfers between processor pairs (12), such that the plurality of pairs together define a closed loop. The present invention enables a processor to automatically balance its workload with other similar processors connected to it, with minimal interprocessor connection.

WO 01/88696 A2

1 Processor with load balancing

2

3 The present invention relates to a system intended for  
4 use in multi-processor computers and in particular to  
5 work load balancing in dataflow parallel computers.

6

7 Multi-processor computers are used to execute programs  
8 that can utilise parallelism, with concurrent work being  
9 distributed across the processors to improve execution  
10 speeds.

11

12 The dataflow model is convenient for parallel execution,  
13 having execution of an instruction either on data  
14 availability or on data demand, not because it is the  
15 next instruction in a list. This also implies that the  
16 order of execution of operations is irrelevant,  
17 indeterminate and cannot be relied upon. The dataflow  
18 model is also convenient for parallel execution because  
19 tokens may flow to specified instructions rather than  
20 having the data stored in a register or memory  
21 potentially accessible by all other instructions.

22

23 In multithreaded dataflow, memory may be introduced into  
24 the flow of tokens to instructions. Only one token is  
25 required to trigger execution of an instruction, the  
26 second operand being fetched from the memory when the  
27 instruction is issued or executed (Coleman, J.N.; A High  
28 Speed Dataflow Processing Element and Its Performance  
29 Compared to a von Neumann Mainframe, Proc. 7<sup>th</sup> IEEE  
30 International Parallel Processing Symposium, California,  
31 pp.24-33, 1993 and Papadopoulos, G.M.; Traub, K.R.;  
32 Multithreading: A Revisionist View of Dataflow  
33 Architectures, Ann. Int. Symp. Comp. Arch., pp.342-351,

1 1991). The result is passed along an arc to initiate a  
2 new instruction and optionally written back to memory.  
3 The memory makes it difficult to avoid side-effects in  
4 hardware, but their problems can be avoided in software  
5 through suitable programming discipline. This  
6 modification of the dataflow model overcomes some of the  
7 physical and speed difficulties of other solutions. In  
8 particular it removes the need for hardware token  
9 matching. As the smallest element that can be  
10 parallelised is a thread, rather than an instruction, the  
11 number of times that the token matching need be performed  
12 is much reduced and so the overheads incurred in  
13 performing the operation in software can be justified.

14  
15 Load balancing in a multi-processor computer has the aim  
16 of ensuring every processor performs an equal amount of  
17 work. This is important for maximising computational  
18 speeds. Traditionally, multi-processor computers have  
19 required complicated hardware or software to perform this  
20 task, and the configuration (i.e., interconnection) of  
21 the processors and memories need to be taken into  
22 account. The load balancing mechanism has greatest  
23 performance restricting effect during times of explosive  
24 parallelism. It must be able to transfer loads  
25 throughout the system quickly, in order to maintain a  
26 higher overall efficiency.

27  
28 Traditional methods of load balancing require expensive  
29 networks and complicated load analysis, and static off-  
30 line scheduling has been used to solve the problem (this  
31 entails analysing the program before it is run to find  
32 out what resources it needs, when, and scheduling all  
33 tasks prior to running).

1

2 On-line load balancing is difficult because of the  
3 complexity and cost in the networks involved. For  
4 example, in a system containing 100 processors, load  
5 balancing potentially requires not only a check of all  
6 100 processors to find out which are free to do work, but  
7 also consideration of which piece of work is best suited  
8 to each processor, depending on what is already scheduled  
9 for that processor. If pieces of work differ in size  
10 then care must be taken to ensure that work is evenly  
11 distributed.

12

13 The difficulty in balancing load is proportional to the  
14 square of the number of processors. If it is decided  
15 that all work must be scheduled within a fixed amount  
16 time, even under the worst case conditions, then because  
17 work can originate anywhere and be scheduled to any  
18 destination, it is necessary to have a network with a  
19 band width proportional to  $N^2$  where  $N$  is the number of  
20 processors. This means that a system with one thousand  
21 processors is ten thousand times more complicated and  
22 costly than a system with only ten processors, despite  
23 having only one hundred times the power. It is desirable  
24 to have a system where complexity and cost are  
25 proportional only to  $N$ , even under worst case conditions.

26

27 In the prior art inventions are known which provide  
28 systems for load balancing in multi-processor computer  
29 systems. US Patent 5,630,129 to Sandia Corporation  
30 describes an application level method for dynamically  
31 maintaining global load balance on a parallel computer.  
32 Global load balancing is achieved by overlapping



1 neighbourhoods of processors, where each neighbourhood  
2 performs local load balancing.

3

4 US Patent 5,701,482 to Hughes Aircraft Company describes  
5 a modular array processor architecture with a control bus  
6 used to keep track of available resources throughout the  
7 architecture under control of a scheduling algorithm that  
8 reallocates tasks to available processors based on a set  
9 of heuristic rules to achieve the load balancing.

10

11 US Patent 5,898,870 to Hitachi, Ltd. describes a load  
12 sharing method of a parallel computer system which sets  
13 resource utilisation target values by work for the  
14 computers in a computer group. Newly requested work  
15 processes are allocated to computers in the computer  
16 group on the basis of the differences between the  
17 resource utilisation target parameter values and current  
18 values of a parameter indicating the resource  
19 utilisation.

20

21 It is an object of the present invention to provide a  
22 processor which can automatically balance its workload  
23 with other similar processors connected to it.

24

25 According to the first aspect of this invention, there is  
26 provided a multi-processor system comprising a plurality  
27 of processors, a plurality of comparison means for  
28 comparing the load at a pair of processors and a  
29 plurality of load balancing means responsive to the  
30 comparison means for passing workload between the said  
31 pair of processors, characterised in that the plurality  
32 of load balancing means defines a closed loop around  
33 which workload can be passed.

1

2 Preferably the passing of workload is uni-directional  
3 around the closed loop.

4

5 More preferably, the passing of workload comprises the  
6 passing of a processing thread.

7

8 Preferably the passing of a processor thread comprises  
9 the passing of an instruction.

10

11 Preferably the passing of an instruction comprises the  
12 passing of an instruction and the pointer to the context  
13 of said instruction.

14

15 According to a second aspect of this invention, there is  
16 provide a method of distributing load among processors in  
17 a multi-processor system. The method comprising the  
18 steps of:

- 19     • comparing the load in pairs of processors and  
20     • transferring workload between said processors  
21 characterised in that the workload is transferred through  
22 a plurality of transfers between pairs, such that the  
23 plurality of pairs together define a closed loop.

24

25 Preferably, the pairs in the closed loop comprising a  
26 first processor and a second processor, the first  
27 processor informs the second processor of the first  
28 processor's workload.

29

30 Preferably, the second processor compares the first  
31 processor's workload with its own workload.

32

1 More preferably, the second processor determines whether  
2 it will request more work from the first processor.

3

4 Preferably, the second processor requests work from the  
5 first processor.

6

7 Optionally, comparison means for comparing the load of  
8 two processors and load balancing means responsive to the  
9 comparison means can be introduced cutting across the  
10 loop to accelerate load balancing around the loop.

11

12 The load balancing means responsive to the comparison  
13 means ensure that between every pair there is a balance  
14 of workload, and a closed loop ensures that every  
15 processor in every pair is downstream of another  
16 processor, which in turn ensures that the entire loop is  
17 inherently balanced with respect to workload.

18

19 With a bi-directional link between the first and second  
20 processor, both processors in a pair inform each other of  
21 workload and request work as appropriate. There is no  
22 requirement for such pairs to be arranged in a circle.

23

24 When work is requested from a processor, preferably that  
25 processor picks up a suitable instruction out of its  
26 pipeline, and transfers that instruction and its context  
27 (e.g., data tokens on input/output arcs) across to the  
28 requesting processor which then inserts it directly into  
29 its own pipeline. This is possible because each  
30 instruction is an independent unit of work within each  
31 processor, and therefore within the system as a whole.

32

1 In order to provide a better understanding of the present  
2 invention an example will now be described, by way of  
3 example only, and with reference to the accompanying  
4 Figures, in which :

5

6 Figures 1 to 3 illustrate configurations of the  
7 processors and workflow in the system of the present  
8 invention

9

10 Figure 4 illustrates a block diagram of the system  
11 including processors and memory

12

13 Figure 5 illustrates thread transfer between a pair  
14 of processors

15

16 The invention is a multi-processor dataflow computer  
17 which functions to balance workload between the  
18 processors.

19

20 Although the embodiments of the invention described with  
21 reference to the drawings comprise computer apparatus and  
22 processes performed in computer apparatus, the invention  
23 also extends to computer programs, particularly computer  
24 programs on or in a carrier, adapted for putting the  
25 invention into practice. The program may be in the form  
26 of source code, object code, a code of intermediate  
27 source and object code such as in partially compiled form  
28 suitable for use in the implementation of the processes  
29 according to the invention. The carrier may be any  
30 entity or device capable of carrying the program.

31

32 For example, the carrier may comprise a storage medium,  
33 such as ROM, for example a CD ROM or a semiconductor ROM,

1 or a magnetic recording medium, for example, floppy disc  
2 or hard disc. Further, the carrier may be a  
3 transmissible carrier such as an electrical or optical  
4 signal which may be conveyed via electrical or optical  
5 cable or by radio or other means.

6  
7 When the program is embodied in a signal which may be  
8 conveyed directly by a cable or other device or means,  
9 the carrier may be constituted by such cable or other  
10 device or means.

11  
12 Alternatively, the carrier may be an integrated circuit  
13 in which the program is embedded, the integrated circuit  
14 being adapted for performing, or for use in the  
15 performance of, the relevant processes.

16  
17 Referring firstly to Figure 1, a closed loop 10 of  
18 processors 11 are connected by link means 12. Preferably  
19 the link means comprises connection through an electrical  
20 circuit or a packet switched network. The link means  
21 provide the means for comparison of workload and passing  
22 of workload between processors. In Figure 1 the link  
23 means 10 are uni-directional, wherein the transfer of  
24 workload through the link means is in one direction.  
25 With a uni-directional link from processor A 13  
26 ("upstream") to processor B 14 ("downstream"), A informs  
27 B of how much workload it has, B then compares this with  
28 its own level of workload, and if B is less loaded than  
29 A, then it requests work from A. It is therefore ensured  
30 that B has at least as much work as A. Such pairs are  
31 linked end to end in a chain, with all the links going in  
32 the same direction, with the ends of the chain joined  
33 together. This forms a closed loop with all the workload

1 transfers travelling in the same direction. Since in  
2 each pair the one downstream of the link has at least as  
3 much work as the one upstream, and every processor in  
4 every pair downstream of another processor, it ensures  
5 that the entire ring is inherently balanced.

6  
7 Referring to Figure 2, a closed loop 20 of processors 21  
8 with bi-directional link means 22 is shown, wherein the  
9 transfer of workload through the link means between each  
10 processor pair is in one direction. The two processors in  
11 a pair both inform each other and request workload as  
12 appropriate.

13  
14 Referring to Figure 3, a closed loop 30 of processors 31  
15 is shown with additional links 32 between pairs cutting  
16 across the ring, which have been introduced to accelerate  
17 load balancing around the ring.

18  
19 Referring to Figure 4, a block diagram of a multi-  
20 processor system 40 is shown, which is a shared memory  
21 multi-processor dataflow computer. The three main  
22 components are processors 41, crossbar switches 42 for  
23 providing the means for relaying memory requests from  
24 processors to memory controllers, and memory controllers  
25 43. We envisage these component being implemented on  
26 separate chips and connected accordingly. Preferably,  
27 the processors are connected in a uni-directional  
28 circular pipeline or closed loop, and access is set as  
29 interleaved memory modules through a crossbar switch  
30 array. Preferably processors issue memory requests to  
31 the crossbar switches, which then relay them to the  
32 memory leaves. Memory controllers return the result of  
33 the request back to the processors via the crossbar

1 switches. Preferably all communication is handled  
2 automatically in hardware. Preferably, inter-processor  
3 communication is invisible to the programmer and program  
4 and preferably comprises load balancing traffic.  
5 Transactions allow several memory accesses to be  
6 performed concurrently; the processor can send out a  
7 stream of requests, those that go back to different  
8 crossbar switches will be handled simultaneously, and the  
9 results will stream back. This reduces rather than just  
10 hides the memory latency, but it is dependent on all  
11 memory leaves being evenly utilised.

12  
13 Each processor keeps track of how many threads it is  
14 hosting at any one time. It passes this information on  
15 to the next processor round the closed loop. This means  
16 that each processor can determine its own load, as well  
17 as the load of its predecessor. By comparing the two  
18 loads, a load imbalance can be calculated. If this is  
19 outside tolerances (e.g., greater than one thread  
20 difference), then the processor may request load from its  
21 predecessor.

22  
23 Referring to Figure 5, a thread transfer between a pair  
24 of processors 50 is shown. Upon receiving a request for  
25 a load, preferably a processor's 51 multiplexer stage 52  
26 will pick out the next passing eligible instruction and  
27 route it out of the input/output unit, IO unit 53.  
28 Preferably, the IO unit 53 comprises a shift register  
29 which transfers the instruction and its flow operands out  
30 to the requesting processor 54 over a thread transfer bus  
31 55. Preferably, the requesting processor 54 accumulates  
32 the transmission in its own IO unit 56 and, when this  
33 shift register is full, the register contents are passed

1 to the multiplexer 57, which then merges it into the  
2 pipeline flow. Preferably, this activity is entirely  
3 invisible to the program.

4

5 Further modification and improvements may be added  
6 without departing from the scope of the invention herein  
7 described.

8



1   Claims

2

3   1.   A multi-processor system comprising a plurality of  
4       processors, a plurality of comparison means for  
5       comparing the load at a pair of processors, and a  
6       plurality of load balancing means responsive to the  
7       comparison means for passing workload between said  
8       pair of processors, characterised in that the  
9       plurality of load balancing means defines a closed  
10      loop around which workload can be passed.

11

12   2.   A system as claimed in claim 1 wherein the passing  
13       of workload is uni-directional around the closed  
14       loop.

15

16   3.   A system as claimed in claims 1 to 2 wherein the  
17       passing of workload comprises the passing of a  
18       processing thread.

19

20   4.   A system as claimed in claim 3 wherein the passing  
21       of a processing thread comprises the passing of an  
22       instruction.

23

24   5.   A system as claimed in claim 4 wherein the passing  
25       of an instruction comprises the passing of an  
26       instruction and a pointer to the context of said  
27       instruction.

28

29   6.   A system as claimed in claims 1 to 5 wherein there  
30       are load balancing means responsive to comparison  
31       means comparing the load of a pair of processors in  
32       the closed loop of claim 1, the said pair of  
33       processors not being compared in claim 1.

1

2 7. A method for distributing load among processors in a  
3 multi-processor system, the method comprising the  
4 steps of:

5

6 Comparing the load in pairs of processors and

7

8 Transferring work load between said processors

9

10 characterised in that the workload is transferred  
11 through a plurality of transfers between pairs of  
12 processors, such that the plurality of pairs  
13 together define a closed loop.

14

15 8. A method as claimed in claim 7 wherein the pairs  
16 comprise a first processor and a second processor,  
17 and first processor informs the second processor of  
18 the first processor's work load.

19

20 9. A method as claimed in claim 8 wherein the second  
21 processor compares the first processor's work load  
22 with its own work load.

23

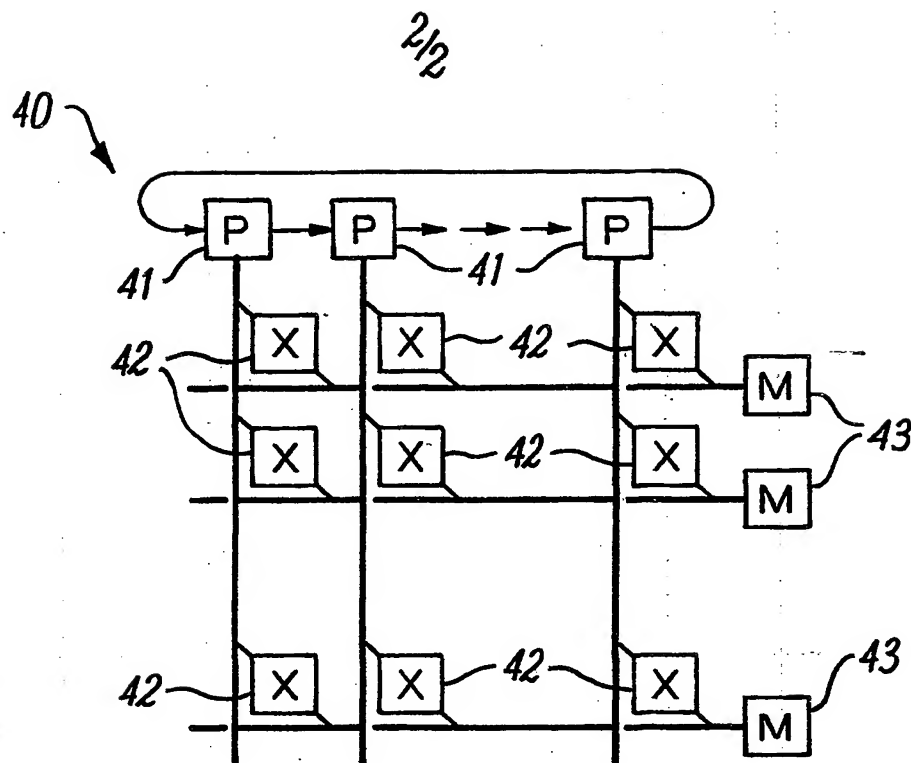
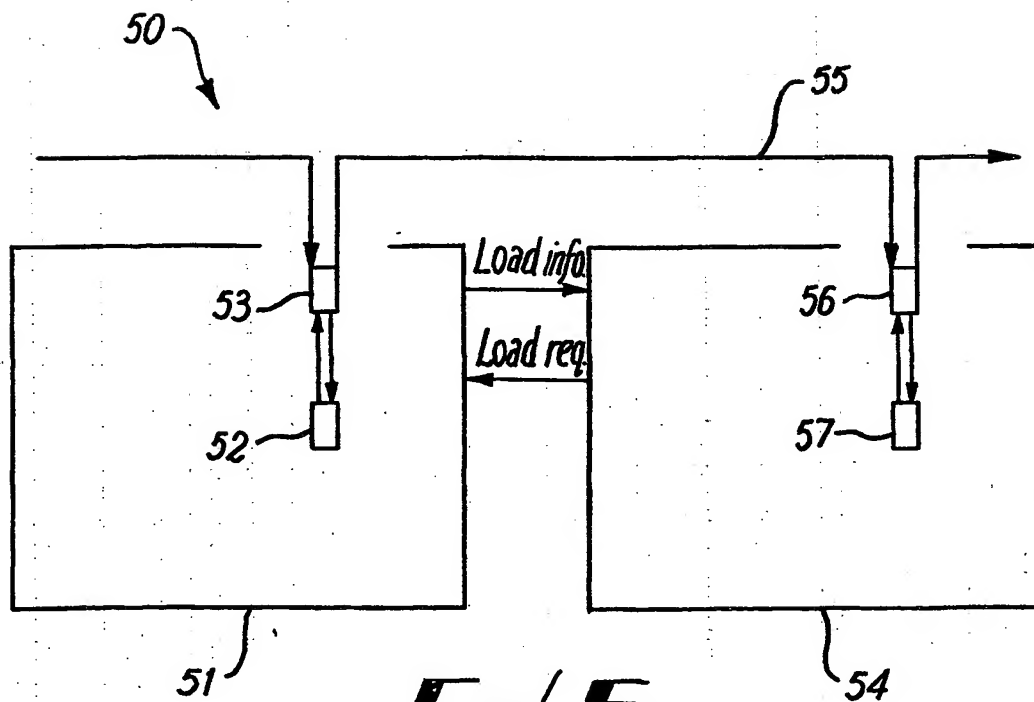
24 10. A method as claimed in claims 8 to 9 wherein the  
25 second processor determines whether it will request  
26 more work from the first processor.

27

28 11. A method as claimed in claims 8 to 10 wherein the  
29 second processor requests work from the first  
30 processor.

31



**FIG. 4****FIG. 5**